

```

1  /*
2  TinyTrackGPS.cpp - A simple track GPS to SD card logger.
3  TinyTrackGPS v0.14
4
5  Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
6  All rights reserved.
7
8  raphael.reyes.carmona@gmail.com
9
10 This file is part of TinyTrackGPS.
11
12 TinyTrackGPS is free software: you can redistribute it and/or modify
13 it under the terms of the GNU General Public License as published by
14 the Free Software Foundation, either version 3 of the License, or
15 (at your option) any later version.
16
17 TinyTrackGPS is distributed in the hope that it will be useful,
18 but WITHOUT ANY WARRANTY; without even the implied warranty of
19 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
20 GNU General Public License for more details.
21
22 You should have received a copy of the GNU General Public License
23 along with TinyTrackGPS. If not, see <https://www.gnu.org/licenses/>.
24 */
25
26 /*****
27 / Programa de localizacion por gps que graba las posiciones en
28 / un fichero de texto cada segundo, de forma diaria.
29 /
30 / - Conectar módulo SD con pin CS (naranja) en pin 10 arduino.
31 /
32 / Uso de librería TinyGPS.
33 / Requiere uso de librería SoftwareSerial, se presupone que disponemos
34 / de un dispositivo GPS serie de 9600-bauds conectado en pines 9(rx) y 8(tx).
35 / - Conectar módulo NMEA-6M (gps) pines 8,9 (9 - pin rx negro)
36 /
37 / - Conectar LCD 16x2 pines 2,3,4,5,6,7 (2-amarillo , 3-azul,
38 / 4-rojo, 5-azul oscuro, 6-verde, 7-blanco)
39 /
40 / - Conectar OLED 0.96" en SDA y SCL. pines A4 y A5 del Arduino UNO.
41 *****/
42
43 // Include libraries.
44 #include <Arduino.h>
45 #include "config.h"
46 #include "Display.h"
47 // #include <SoftwareSerial.h>
48 #include "TinyGPS_GLONASS_fixed.h"
49 #if defined(__LGT8F__)
50 #include <LowPower.h>
51 #endif
52 #include "SdFat.h"
53 #include "Vcc.h"
54 #include <sdios.h>
55 #include <UTMConversion.h>
56 #include <Timezone.h>
57 #if defined(TIMEZONE_FILE)
58 #include "ConfigFile.h"
59 #endif
60 #include "Semphr.h"
61

```

```

62 // Definimos el Display
63 #if defined(DISPLAY_TYPE_LCD_16X2)
64 Display LCD(LCD_16X2);
65 #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
66 Display LCD(LCD_16X2_I2C);
67 #elif defined(DISPLAY_TYPE_SDD1306_128X64)
68 Display LCD(SDD1306_128X64);
69 #elif defined(DISPLAY_TYPE_SH1106_128X64)
70 Display LCD(SDD1306_128X64);
71 #elif defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
72 Display LCD(SDD1306_128X64);
73 #else
74 #define NO_DISPLAY
75 #include <LowPower.h>
76 #endif
77
78 // Chip select may be constant or RAM variable.
79 const uint8_t SD_CS_PIN = 10;
80 // Pin numbers in templates must be constants.
81 const uint8_t SOFT_MISO_PIN = 12;
82 const uint8_t SOFT_MOSI_PIN = 11;
83 const uint8_t SOFT_SCK_PIN = 13;
84
85 // SdFat software SPI template
86 SoftSpiDriver<SOFT_MISO_PIN, SOFT_MOSI_PIN, SOFT_SCK_PIN> softSpi;
87 // Speed argument is ignored for software SPI.
88 #if ENABLE_DEDICATED_SPI
89 #define SD_CONFIG SdSpiConfig(SD_CS_PIN, DEDICATED_SPI, SD_SCK_MHZ(0), &softSpi)
90 #else // ENABLE_DEDICATED_SPI
91 #define SD_CONFIG SdSpiConfig(SD_CS_PIN, SHARED_SPI, SD_SCK_MHZ(0), &softSpi)
92 #endif // ENABLE_DEDICATED_SPI
93 SdFat card; //SdFat.h library.
94 File file;
95 bool SDReady;
96 bool SaveOK;
97
98 // Variables y clases para obtener datos del GPS y conversion UTM.
99 TinyGPS gps;
100 GPS_UTM utm;
101 //SoftwareSerial gps_serial(9, 8);
102 #define gps_serial Serial // Uses Serial to read GPS info.
103 int year_gps;
104 //byte month_gps, day_gps, hour_gps, minute_gps, second_gps;
105 float flat, flon;
106 unsigned long age;
107 unsigned int elev;
108
109 // Variables para configurar Timezone.
110 #ifndef TIMEZONE_FILE
111 // Central European Time (Frankfurt, Paris) See below for other zone.
112 TimeChangeRule CEST = {"CEST", Last, Sun, Mar, 2, 120}; // Central European
Summer Time
113 TimeChangeRule CET = {"CET ", Last, Sun, Oct, 3, 60}; // Central European
Standard Time
114 Timezone CE(CEST, CET);
115 #define TimeZone CE
116 #else
117 TimeChangeRule UT = {"UTC", Last, Sun, Mar, 1, 0}; // UTC
118 TimeChangeRule UST;
119 Timezone TimeZone(UT);
120

```

```

121 // Loads the configuration from a file
122 bool loadConfiguration(TimeChangeRule *UST, TimeChangeRule *UT) {
123
124     boolean file;
125     uint8_t read;
126     ConfigFile<12> TimeConf;
127
128     if((file = TimeConf.begin("Time.cfg"))){
129         read = 0;
130         while(TimeConf.readNextSetting()){
131
132             char opt[5];
133             strcpy(opt, TimeConf.getName());
134
135             if (!strcmp(opt, "USTw")) {
136                 read++;
137                 UST->week = TimeConf.getIntValue();
138             }
139             else if (!strcmp(opt, "USTd")) {
140                 read++;
141                 UST->dow = TimeConf.getIntValue();
142             }
143             else if (!strcmp(opt, "USTm")) {
144                 read++;
145                 UST->month = TimeConf.getIntValue();
146             }
147             else if (!strcmp(opt, "USTh")) {
148                 read++;
149                 UST->hour = TimeConf.getIntValue();
150             }
151             else if (!strcmp(opt, "USTo")) {
152                 read++;
153                 UST->offset = TimeConf.getIntValue();
154             }
155
156             else if (!strcmp(opt, "UTw")) {
157                 read++;
158                 UT->week = TimeConf.getIntValue();
159             }
160             else if (!strcmp(opt, "UTd")) {
161                 read++;
162                 UT->dow = TimeConf.getIntValue();
163             }
164             else if (!strcmp(opt, "UTm")) {
165                 read++;
166                 UT->month = TimeConf.getIntValue();
167             }
168             else if (!strcmp(opt, "UTh")) {
169                 read++;
170                 UT->hour = TimeConf.getIntValue();
171             }
172             else if (!strcmp(opt, "UTO")) {
173                 read++;
174                 UT->offset = TimeConf.getIntValue();
175             }
176             /*
177             // Put a nameIs() block here for each setting you have.
178             //if(TimeConf.nameIs("USTabbre"))
179             //    strcpy(UST.abbrev, "UST");
180
181             if(TimeConf.nameIs("USTw"))

```

```

182     UST->week = TimeConf.getIntValue();
183     else if(TimeConf.nameIs("USTd"))
184         UST->dow = TimeConf.getIntValue();
185     else if(TimeConf.nameIs("USTm"))
186         UST->month = TimeConf.getIntValue();
187     else if(TimeConf.nameIs("USTh"))
188         UST->hour = TimeConf.getIntValue();
189     else if(TimeConf.nameIs("USTo"))
190         UST->offset = TimeConf.getIntValue();
191
192     //else if(TimeConf.nameIs("UTabbre"))
193     //    strcpy(UST.abbrev,"UT");
194     else if(TimeConf.nameIs("UTw"))
195         UT->week = TimeConf.getIntValue();
196     else if(TimeConf.nameIs("UTd"))
197         UT->dow = TimeConf.getIntValue();
198     else if(TimeConf.nameIs("UTm"))
199         UT->month = TimeConf.getIntValue();
200     else if(TimeConf.nameIs("UTh"))
201         UT->hour = TimeConf.getIntValue();
202     else if(TimeConf.nameIs("UTo"))
203         UT->offset = TimeConf.getIntValue();
204     */
205     strcpy(UST->abbrev,"UST");
206     strcpy(UT->abbrev,"UT");
207 }
208 }
209 TimeConf.end();
210
211 //Serial.print(UST->offset);
212 //Serial.println(UST->abbrev);
213 //Serial.print(UT->offset);
214 //Serial.println(UT->abbrev);
215
216 if(read == 10) return true;
217 return false;
218 }
219 #endif
220 // Variables para gestionar el tiempo local.
221 TimeElements time_gps;
222 time_t utctime;
223 time_t localtime;
224 time_t prevtime;
225
226 //-----
227 /*
228  * User provided date time callback function.
229  * See SdFile::dateTimeCallback() for usage.
230  */
231 void dateTime(uint16_t* date, uint16_t* time) {
232     // User gets date and time from GPS or real-time
233     // clock in real callback function
234
235     // return date using FAT_DATE macro to format fields
236     /*date = FAT_DATE(year, month, day);
237     *date = (year(localtime)-1980) << 9 | month(localtime) << 5 | day(localtime);
238
239     // return time using FAT_TIME macro to format fields
240     /*time = FAT_TIME(hour, minute, second);
241     *time = hour(localtime) << 11 | minute(localtime) << 5 | second(localtime) >> 1;
242 }

```

```

243 //-----
244
245 #ifndef NO_DISPLAY
246 #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
247 bool pinswitch();
248 #endif
249 #endif
250 //void GPSRefresh();
251 #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
252 unsigned long iteration = 0;
253 #endif
254
255 #define BAT_MIN 3.500
256 #define BAT_MAX 4.250
257 #define BAT_MIN_mV 3500
258 #define BAT_MAX_mV 4250
259 #define ALFA_BAT 1.0e2 // 100 / (BAT_MAX - BAT_MIN) -> 0..100%
260 #define BETA_BAT 2.5e1 // ALFA_BAT / 4 -> 0..25
261
262 Vcc vcc(1.0);
263
264 uint8_t charge_level(){
265     //float f_charge = (vcc.Read_Volts() * BETA_BAT) - (BAT_MIN * BETA_BAT);
266     //int i_charge = (int)f_charge;
267     //uint8_t charge = constrain(i_charge, 0, 26);
268     //return charge;
269     //float f_charge = vcc.Read_Perc(BAT_MIN,BAT_MAX);
270     //int i_charge = (int)f_charge;
271     //return (i_charge >> 2);
272     uint16_t volt = vcc.Read_Volts_fast();
273     uint16_t charge = map(vcc.Read_Volts_fast(),BAT_MIN_mV,BAT_MAX_mV,0,25);
274     if(volt < BAT_MIN_mV) return 0;
275     return (constrain(charge,0,25));
276 }
277
278 bool GPSData(TinyGPS &gps, GPS_UTM &utm) {
279     static bool save = false;
280     char GPSLogFile[13];
281
282     sprintf(GPSLogFile, "%04d%02d%02d.csv", year(localtime), month(localtime),
day(localtime));
283
284     //SdFile::dateTimeCallback(dateTime);
285     FsDateTime::setCallback(dateTime);
286
287     // Si no existe el fichero lo crea y añade las cabeceras.
288     if (SDReady && !card.exists(GPSLogFile)) {
289         if (file.open(GPSLogFile, O_CREAT | O_APPEND | O_WRITE)) {
290             //Serial.print(F("New GPSLogFile, adding heads..."));
291             file.println(F("Time,Latitude,Longitude,Elevation,UTM Coords(WGS84)"));
292             //Serial.println(F("Done."));
293             file.close();
294         }
295         //else {
296         //Serial.println(F("** Error creating GPSLogFile. **"));
297         //}
298     }
299     if (SDReady && (file.open(GPSLogFile, O_APPEND | O_WRITE))) {
300         //Serial.print(F("Open GPSLogFile to write..."));
301         char str[19];
302         char comma = 0x2c;

```

```

303
304     sprintf(str, "%02d:%02d:%02d", hour(localtime), minute(localtime),
second(localtime));
305     file.print(str);
306     file.print(comma);
307     file.print(flat,6);
308     file.print(comma);
309     file.print(flon,6);
310     file.print(comma);
311     file.print(elev);
312     file.print(comma);
313     sprintf(str, "%02d%c %ld %ld", utm.zone(), utm.band(), utm.X(), utm.Y());
314     file.print(str);
315     file.print("\n");
316     file.close();
317     save = true;
318     //Serial.println(F("Done."));
319 } //else {
320     //Serial.println(F("*** Error opening GPSLogFile. ***"));
321 //}
322 //} //else Serial.println(F("*** GPS signal lost. ***"));
323 return (save && SDReady);
324 }
325
326 #ifndef NO_DISPLAY
327 void ScreenPrint(Display &LCD, TinyGPS &gps, GPS_UTM &utm){
328
329     unsigned short sats;
330
331     sats = gps.satellites();
332     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
333         bool print_utm = false;
334         bool print_grades = false;
335
336         if (!pinswitch()) print_utm = true;
337         else print_grades = true;
338
339         if (print_utm) {
340             #endif
341             char line[12];
342             #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
343             sprintf(line, "%02d%c?%ld?", utm.zone(), utm.band(), utm.X());
344             #else
345             sprintf(line, "%02d%c %ld ", utm.zone(), utm.band(), utm.X());
346             #endif
347             //Serial.println(line);
348             LCD.print(0,0,line);
349             LCD.print_PChar((byte)6);
350             #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
351             sprintf(line, "%02hu?", sats);
352             #else
353             sprintf(line, "%02hu ", sats);
354             #endif
355             //Serial.println(line);
356             LCD.print(12,0,line);
357             (SaveOK) ? LCD.print_PChar((byte)7) : LCD.print("-");
358
359             // New line
360             #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
361             sprintf(line, "%ld?", utm.Y());
362             #else

```

```

363     sprintf(line, "%ld ", utm.Y());
364 #endif
365 //Serial.println(line);
366 LCD.print(1,1,line);
367 LCD.print_PChar((byte)5);
368 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
369     sprintf(line, "%u@", elev);
370 #else
371     sprintf(line, "%um", elev);
372 #endif
373 //Serial.println(line);
374
375     unsigned int elev_n = elev;
376     byte n = 1;
377     while (elev_n > 9){
378         elev_n /= 10;
379         n++;
380     }
381     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
382     for(byte i = 5-n; i>0; i--) LCD.print(9+i,1," ");
383     #elif defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
384     for(byte i = 5-n; i>0; i--) LCD.print(9+i,1,"?");
385     #endif
386     LCD.print(15-n,1,line);
387
388     /*
389     if (elev < 10) LCD.print(14,1,line);
390     else if (elev < 100) LCD.print(13,1,line);
391     else if (elev < 1000) LCD.print(12,1,line);
392     else LCD.print(11,1,line);
393     */
394     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
395     }
396
397     if (print_grades) {
398         static char line[12];
399     #endif
400     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
401     LCD.print(0, 2, ")?");
402     #else
403     LCD.print(1,(LCD.display_type() == SDD1306_128X64) ? 2 : 0,"LAT=");
404     #endif
405     dtostrf(flat, 8, 6, line);
406     LCD.print(line);
407
408     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
409     LCD.print(0, 3,"*?");
410     #else
411     LCD.print(1,(LCD.display_type() == SDD1306_128X64) ? 3 : 1,"LON=");
412     #endif
413     dtostrf(flon, 8, 6, line);
414     LCD.print(line);
415     }
416 #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
417 }
418
419 bool pinswitch() {
420     static bool prevpin = 0;
421     static bool pin = 0;
422     unsigned long time;
423     time = millis();

```

```

424
425     pin = bitRead(time,13); // Change every 8192 miliseconds.
426
427     if (prevpin^pin) LCD.clr(); // Clear display when change between modes.
428
429     return pin;
430 }
431 #endif
432 #endif
433
434 inline void set_time(){
435     //static TimeElements time_gps;
436
437     time_gps.Year = year_gps - 1970;
438     //time_gps.Month = month_gps;
439     //time_gps.Day = day_gps;
440     //time_gps.Hour = hour_gps;
441     //time_gps.Minute = minute_gps;
442     //time_gps.Second = second_gps;
443
444     utctime = makeTime(time_gps);
445     localtime = TimeZone.toLocal(utctime);
446 }
447
448 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
449 Semphr semaphore;
450
451 void drawBatteryIcon(){
452     LCD.drawbattery(charge_level());
453 }
454 #endif
455
456 void setup(void) {
457     #if defined(__LGT8F__)
458     ECCR = 0x80;
459     ECCR = 0x00;
460     #endif
461     delay(100);
462     //Serial.begin(9600);
463     gps_serial.begin(9600);
464
465     //Serial.print(F("Initializing SD card..."));
466
467     SDReady = card.begin(SD_CONFIG);
468     //(SDReady) ? Serial.println(F("Done. ")) : Serial.println(F("FAILED!"));
469
470     // Config TimeZone (localtime) with 'Time.cfg' file on SD.
471     #if defined(TIMEZONE_FILE)
472     if(loadConfiguration(&UST,&UT)) TimeZone.setRules(UST,UT);
473     #endif
474
475     /* Iniciaización del display LCD u OLED */
476     #ifndef NO_DISPLAY
477     LCD.start();
478     #endif
479
480     //Serial.print(F("Waiting for GPS signal..."));
481     #ifndef NO_DISPLAY
482     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C) ||
defined(DISPLAY_TYPE_SDD1306_128X64) || defined(DISPLAY_TYPE_SH1106_128X64)
483     LCD.print(NAME, VERSION,"Waiting GPS",UT.abbrev);

```



```

484 #elif defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
485 #if defined(__LGT8F__)
486 LCD.DrawLogo();
487 LCD.print(3,UT.abbrev);
488 #else
489 LCD.print(NAME_M, VERSION,UT.abbrev);
490 #endif
491 #endif
492 unsigned int time = 0;
493 #endif
494
495 for(uint8_t i = 8; i--;) charge_level();
496
497 bool config = false;
498
499 do {
500     if(charge_level() == 0) {
501         LCD.clr();
502         #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
503         drawBatteryIcon();
504         #endif
505         while(charge_level() == 0);
506         setup();
507     }
508     #ifndef NO_DISPLAY
509     LCD.wait_anin(time++);
510     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
511     drawBatteryIcon();
512     #endif
513     #endif
514     for (unsigned long start = millis(); millis() - start < 1000;) {
515         while (gps_serial.available() > 0) {
516             char c = gps_serial.read();
517             //Serial.write(c); // uncomment this line if you want to see the GPS data
518             flowing
519             if (gps.encode(c)) {// Did a new valid sentence come in?
520                 // gps.crack_datetime(&year_gps, &month_gps, &day_gps, &hour_gps,
521                 // &minute_gps, &second_gps, NULL, &age);
522                 gps.crack_datetime(&year_gps, &time_gps.Month, &time_gps.Day,
523                 &time_gps.Hour, &time_gps.Minute, &time_gps.Second, NULL, &age);
524                 (age != TinyGPS::GPS_INVALID_AGE) ? config = true : config = false;
525             }
526         }
527     }
528     }while(!config);
529
530     set_time();
531     prevtime = utctime;
532     //Serial.println(F("Done."));
533     //Serial.println(F("Configuration ended."));
534     #ifndef NO_DISPLAY
535     LCD.clr();
536     #endif
537 }
538
539 void loop(void) {
540     static bool gps_ok = false;
541     static bool needcharge = false;
542     uint8_t charge;
543     uint8_t errorSD;

```

```

542 while (gps_serial.available() > 0) {
543     char c = gps_serial.read();
544     //Serial.write(c); // uncomment this line if you want to see the GPS data
    flowing
545     if (gps.encode(c)) { // Did a new valid sentence come in?
546 //         gps.crack_datetime(&year_gps, &month_gps, &day_gps, &hour_gps, &minute_gps,
&second_gps, NULL, &age);
547         gps.crack_datetime(&year_gps, &time_gps.Month, &time_gps.Day, &time_gps.Hour,
&time_gps.Minute, &time_gps.Second, NULL, &age);
548         (age != TinyGPS::GPS_INVALID_AGE) ? gps_ok = true : gps_ok = false;
549         #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
550             semaphore.set();
551         #endif
552         if(!SDReady)
553             if(card.cardBegin(SD_CONFIG)) SDReady = card.begin(SD_CONFIG);
554     }
555 }
556
557 gps.f_get_position(&flat, &flon, &age);
558 if ((elev = gps.altitude()) == TinyGPS::GPS_INVALID_ALTITUDE) elev = 0;
559 else elev /= 100L;
560 utm.UTM(flat, flon);
561
562 set_time();
563
564 //Serial.println(utctime);
565 //Serial.println(localtime);
566
567 charge = charge_level();
568
569 if (gps_ok && !(needcharge)) {
570     if (utctime > prevtime) {
571         (!(errorSD = card.sdErrorCode())) ? SDReady = true : SDReady = false;
572         if (errorSD == 11) card.end();
573         //Serial.println(errorSD);
574         if (!errorSD) SaveOK = GPSTData(gps, utm);
575         else SaveOK = false;
576         prevtime = utctime;
577         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
578             iteration++;
579         #endif
580     }
581     #ifndef NO_DISPLAY
582     ScreenPrint(LCD, gps, utm);
583     gps_ok = false;
584 } else if (charge==0){
585     LCD.clr();
586     needcharge = true;
587     #endif
588 }
589
590 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
591 if((charge==0) && bitRead(millis(),9))
592     semaphore.set();
593 else if((millis()&0x1fff) == 0x1fff)
594     semaphore.set();
595 semaphore(drawBatteryIcon);
596 #endif
597
598 if(needcharge) (charge > 5) ? needcharge = false : needcharge = true;
599

```

```
600     #if defined(__LGT8F__)
601     LowPower.idle(SLEEP_120MS, ADC_ON, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_ON,
602     USART0_ON, TWI_ON);
603     #endif
604     #ifndef NO_DISPLAY
605     LowPower.idle(SLEEP_120MS, ADC_ON, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_ON,
606     USART0_ON, TWI_ON); // para NO_DISPLAY.
607     #endif
608 }
```